int sets[size][size];

sets[i][j]=-999;

for(int j=0;j<size;j++)

//INITIALIZATION

{

for(int i=0;i<size;i++)

matrix[i][j]=0;}

list[i][j]=0;

for(int j=0;j<3;j++){

for(int i=0;i<size;i++) //Initialize the adjacency matrix

vertices=0,k=0;

{

graph()

public:

int vertices,edges,k;

int matrix[size][size];

**Kruskal Algorithm**

int list[size][3];

{

class graph

using namespace std;

#define size 20

#define infinity 9999

#include<stdlib.h>

#include<stdio.h>

#include<iostream>

**CODE:**

Name : Kunal Gupta

Roll no :231023

Batch :A1

cin>>vertices;

}

k++;

list[k][2]=weight;

list[k][1]=des;

list[k][0]=src;

matrix[src][des]=matrix[des][src]=weight;

cin>>src>>des>>weight;

cout<<"\nEnter an edge (u,v,weight) :";

{

for(i=0;i<edges;i++)

cout<<"\nEnter the list of edges : ";

cin>>edges;

cout<<"\nEnter no of edges :";

sets[i][0]=i;

cout<<"\nEnter no. of vertices :";

int i,j,src,des,weight;

{

void graph:: readgraph()

};

int findset(int);

void merge(int,int);

void kruskal();

void sort();

void printgraph();

void readgraph();

}

}

{

src= list[count][0];

{

while(count<=vertices)

int src,des,set1,set2,count=0;

int cost=0;

sort();

{

void graph:: kruskal()

}

}

}

return i;

if(vertex==sets[i][j])

}

for(int j=0;j<size;j++)

{

for(int i=0;i<size;i++)

{

int graph:: findset(int vertex)

}

cout<<"\n(u,v,weight)= ("<<list[i][0]<<","<<list[i][1]<<","<<list[i][2]<<")";

for(i=0;i<k;i++)

cout<<"\nList of edges in the spanning tree: ";

int i;

{

void graph:: printgraph()

cout<<endl;

}

i++;

sets[n2][i]=-999;

sets[n1][j++]=sets[n2][i];

{

while(sets[n2][i]!=-999)

j++;

while(sets[n1][j]!=-999)

int j=0,i=0;

{

void graph:: merge(int n1,int n2)

}

}

des= list[count][1];

}

cout<<sets[set1][i]<<"\t";

{

for(int i=0;sets[set1][i]!=-999;i++)

count++;

}

cout<<"COST == "<<cost<<endl;

cost+=list[count][2];

merge(set1,set2);

{

if(set1!=set2)

set2=findset(des);

set1=findset(src);

list[j+1][p]=temp;

cout<<"Enter your Choice"<<endl;

do{

int choice;

graph g1;

{

int main()

}

cout<<"\n(u,v,weight)= ("<<list[i][0]<<","<<list[i][1]<<","<<list[i][2]<<")";

for(int i=0;i<k;i++)//Scan the upper triangle of the adjacency matrix

cout<<"After Sorting,we get"<<endl;

}

}

}

}

}

list[j][p]=list[j+1][p];

temp=list[j][p];

{

for(int p=0;p<3;p++)

{

if(list[j][2] > list[j+1][2])

{

for(int j=0;j<k;j++)

{

for(int i=0;i<k;i++)

int temp;

{

void graph:: sort()

break;

Enter no. of vertices :4

1

4. EXIT

3. Find MST using Kruskal

2. Display the graph

1. Read the graph

Enter your Choice

**OUTPUT:**

}

return 0;

}while(choice!=4);

}

cout<<"1. Read the graph"<<endl;

case 3: g1.kruskal();

break;

case 2: g1.printgraph();

break;

case 1: g1.readgraph();

{

switch(choice)

cin>>choice;

cout<<"4. EXIT"<<endl;

cout<<"3. Find MST using Kruskal"<<endl;

cout<<"2. Display the graph"<<endl;

0

4. EXIT

3. Find MST using Kruskal

2. Display the graph

1. Read the graph

Enter your Choice

20

3

Enter an edge (u,v,weight) :1

18

2

Enter an edge (u,v,weight) :0

11

Enter no of edges :6

Enter an edge (u,v,weight) :3

12

3

Enter an edge (u,v,weight) :2

17

2

Enter an edge (u,v,weight) :1

15

1

Enter an edge (u,v,weight) :0

Enter the list of edges :

2

COST == 11

3

0

COST == 23

2

3

0

COST == 38

2

(u,v,weight)= (0,2,18)0

3

3

0

0

1

1

Enter your Choice

1. Read the graph

3. Find MST using Kruskal

List of edges in the spanning tree:

(u,v,weight)= (0,1,15)

(u,v,weight)= (1,2,17)

(u,v,weight)= (2,3,12)

(u,v,weight)= (3,0,11)

(u,v,weight)= (0,2,18)

(u,v,weight)= (1,3,20)Enter your Choice

1. Read the graph

2. Display the graph

2

4. EXIT

3

After Sorting,we get

(u,v,weight)= (0,0,0)

(u,v,weight)= (3,0,11)

(u,v,weight)= (2,3,12)

(u,v,weight)= (0,1,15)

(u,v,weight)= (1,2,17)

2. Display the graph

3. Find MST using Kruskal

4. EXIT

4

\*/